ISO 29990:2010

NATIONAL INFOCOMM COMPETENCY FRAMEWORK

NUS National University of Singapore

ISS INSTITUTE OF SYSTEMS SCIENCE

FUNDAMENTAL OF PROGRAMMING IN C#

INTRODUCTION TO C#

Objectives

- Understand the structure of C# program
- Understand various C# terminology
- Write and execute a simple C# code using command line/ terminal

Agenda

- Introduction to C#
- Compiling and Running C# Program
- Common C# Syntax Errors

What is C#

https://docs.microsoft.com/en-us/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework

- C# is an elegant and type-safe object-oriented language that enables developers to build a variety of secure and robust applications that run on the .NET Framework.
- You can use C# to create Windows client applications, XML Web services, distributed components, client-server applications, database applications, and much, much more.
- Visual C# provides an advanced code editor, convenient user interface designers, integrated debugger, and many other tools to make it easier to develop applications based on the C# language and the .NET Framework.

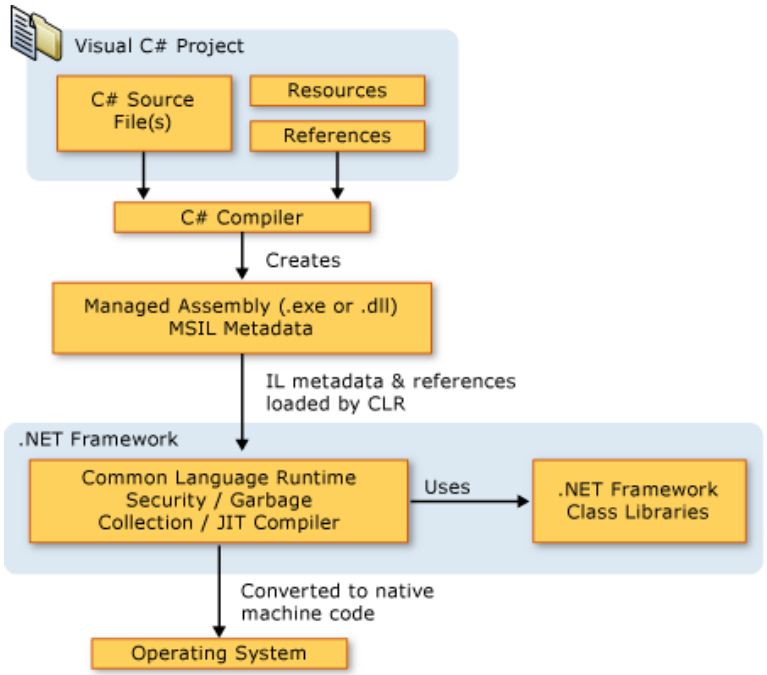
What is C#

- Type safe
  - The programming language is designed to prevent type error such as assigning number value to a date variable

- Object-oriented language
  - This will be covered in more detail in the next module
  - Object-oriented programming:
    - Programming paradigm based on the concepts of objects and classes

.NET Framework

- .NET Framework includes 2 things:
  - A virtual execution system called CLR (common language runtime) that can run CLI code (common language infrastructure)
  - Unified set of class libraries
- There are other languages that can produce programs that runs on CLR and they are interoperable
  - E.g. F#, VB.NET
- There are multiple flavour of .NET Framework
  - .NET Standard (on Windows only)
  - .NET Core (multi-platform)



C# Characteristics to note

- C# is truly object oriented
  - We need to create at least one class when writing a program
- C# is case sensitive
  - Mind your case/capitalization
  - TomAndJerry, tomandjerry, TOMANDJERRY, tomANDjerry are considered different

First C# Program

FirstProgram.cs

```
using System;

namespace FirstProject
{
    /* This program when executed will print
       Welcome to ISS and
       Everyone can program. */

    // An example program for FOPCS
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Welcome to ISS!");
            Console.WriteLine("Everyone can program");
        }
    }
}
```

First C# Program - Main

FirstProgram.cs

```
using System;

namespace FirstProject
{
    /* This program when executed will print
       Welcome to ISS and
       Everyone can program. */

    // An example program for FOPCS
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Welcome to ISS!");
            Console.WriteLine("Everyone can program");
        }
    }
}
```

Program execution starts here.

- `Main` method is the entry point of a C# application
  - .NET will look for this method within a class and execute the method when application starts
  - Main must be static and can return `void` or `int`
    - No difference for our purpose in this module
  - Can be declared with or without `string[]` parameter.

https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/main-and-command-args/

First C# Program - Statements

FirstProgram.cs

```
using System;

namespace FirstProject
{
    /* This program when executed will print
       Welcome to ISS and
       Everyone can program. */

    // An example program for FOPCS
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Welcome to ISS!");
            Console.WriteLine("Everyone can program");
        }
    }
}
```

We write our instructions as statements within the methods

- A method contains a series of statements (instructions) wrapped in a pair of curly braces { }
- The instruction will be executed one-by-one from top to bottom.
- Every statement ends with a semi-colon ;
  - C# will raise an error if you forget your semi-colon

First C# Program - Class

FirstProgram.cs

```
using System;

namespace FirstProject
{
    /* This program when executed will print
       Welcome to ISS and
       Everyone can program. */

    // An example program for FOPCS
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Welcome to ISS!");
            Console.WriteLine("Everyone can program");
        }
    }
}
```

Methods have to be written inside a class.

- A class is a template for an object (TBD in next module)
- A class can contain multiple methods
- The contents of a class must be wrapped inside its curly braces

First C# Program - Comments

FirstProgram.cs

```
using System;

namespace FirstProject
{
    /* This program when executed will print
       Welcome to ISS and
       Everyone can program. */

    // An example program for FOPCS
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Welcome to ISS!");
            Console.WriteLine("Everyone can program");
        }
    }
}
```

Comments should be added to make your program more readable

- Comments are useful for improving readability and documentation
- Single comments are placed after // symbol
- Multi-line comments are enclosed between /* and */

First C# Program - Namespace

FirstProgram.cs

```
using System;

namespace FirstProject
{
    /* This program when executed will print
       Welcome to ISS and
       Everyone can program. */

    // An example program for FOPCS
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Welcome to ISS!");
            Console.WriteLine("Everyone can program");
        }
    }
}
```

Namespace is useful to avoid classes can be referred to uniquely

- To prevent class name duplication, we can name put classes within namespaces
- Namespaces are optional

Namespace	Class	Full class name
	Program	Program
ISS	Program	ISS.Program
NUS.ISS	Program	NUS.ISS.Program

First C# Program - Using

FirstProgram.cs

```
using System;

namespace FirstProject
{
    /* This program when executed will print
       Welcome to ISS and
       Everyone can program. */

    // An example program for FOPCS
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Welcome to ISS!");
            Console.WriteLine("Everyone can program");
        }
    }
}
```

We want to be able to use classes in `System` namespace without having to qualify or specify the namespace

e.g. we can refer to `System.Console` class as just `Console`

- Using directive can appear at the beginning of a source file, before any namespace or type definition
- In any namespace but before any namespace or types declared in this namespace

https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/using-directive

First C# Program

FirstProgram.cs

```
using System;

namespace FirstProject
{
    /* This program when executed will print
       Welcome to ISS and
       Everyone can program. */

    // An example program for FOPCS
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Welcome to ISS!");
            Console.WriteLine("Everyone can program");
        }
    }
}
```

We call a `WriteLine` method to print a text to the screen and then terminate the line

Console.WriteLine Method

Namespace: System

Assemblies: System.Console.dll, mscorlib.dll, netstandard.dll

Writes the specified data, followed by the current line terminator, to the standard output stream.

Overloads

WriteLine(String, Object, Object)	Writes the text representation of the specified objects, followed by standard output stream using the specified format information.
WriteLine(String)	Writes the specified string value, followed by the current line termin
WriteLine(Char[], Int32, Int32)	Writes the specified subarray of Unicode characters, followed by th standard output stream

- In our program, we can use methods in .NET Framework built-in libraries or any additional third-party libraries
- We must refer to the documentation on the method usage

https://docs.microsoft.com/en-us/dotnet/api/system.console.writeline?view=netframework-4.7.2

Statement

WriteLine is a method that belongs to System.Console class

```
System.Console.WriteLine("Welcome to ISS!");
```

`System.Console` class is another way to say that there's a `Console` class in a `System` namepace

We can refer to the class just by typing Console as long as we have import System class with "`using System;`" directive

Method can have zero to multiple arguments

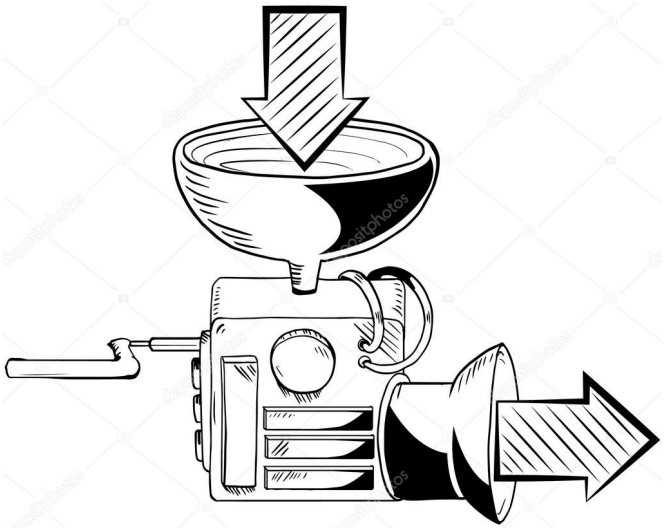
Arguments are specified within the bracket separated with commas

In this example, there's only one argument which contains the text that we want to print out.

We need to enclose text (a.k.a. string) value with double quote in C#

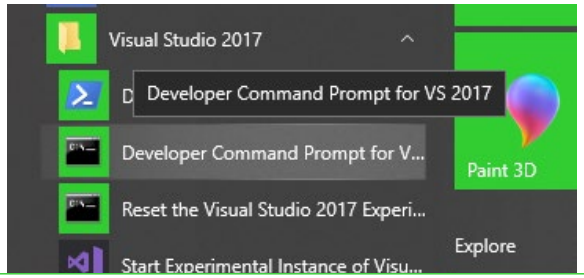
Method

- A metaphor for a method is like a machine that can take multiple input and produce an output
- Arguments are the input to the machine
- The output is the return value
  - Some methods like WriteLine doesn't have a return value



Compiling

- C# compiler transform C# source code into CLI (common language infrastructure) code that can be executed by .NET Framework.
- Run Developer Command Prompt from Start Menu
- Go to the folder where we put our source code
- Run csc (C Sharp Compiler) and pass our source code file
- This is probably the first and last time we use csc.exe manually in this module



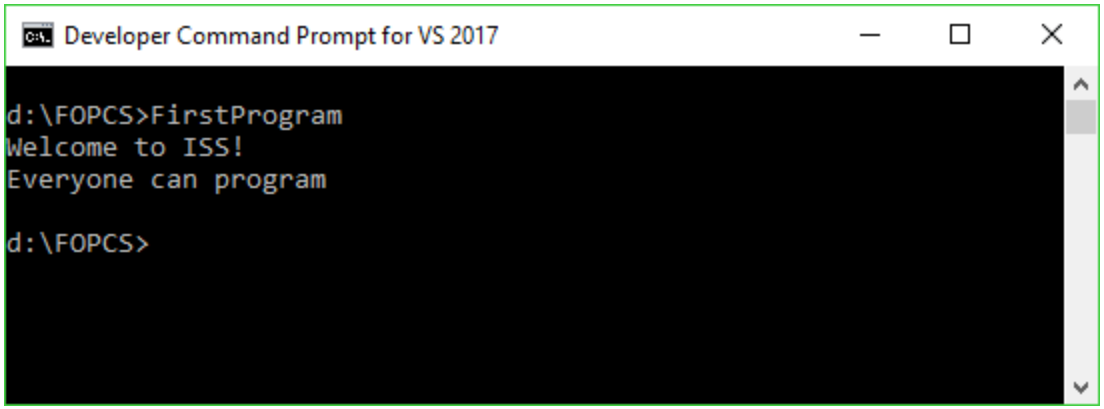
Developer Command Prompt for VS 2017

```
D:\>cd d:\FOPCS

d:\FOPCS>dir
 Volume in drive D is DATA
 Volume Serial Number is B200-6E0E

 Directory of d:\FOPCS

08/02/2019  04:25 PM    <DIR>          .
08/02/2019  04:25 PM    <DIR>          ..
08/02/2019  04:25 PM               412 FirstProgram.cs
               1 File(s)            412 bytes
               2 Dir(s)  701,679,104,000 bytes free

d:\FOPCS>csc FirstProgram.cs
Microsoft (R) Visual C# Compiler version 2.8.3.62923 (7aafab56)
Copyright (C) Microsoft Corporation. All rights reserved.


d:\FOPCS>dir
 Volume in drive D is DATA
 Volume Serial Number is B200-6E0E

 Directory of d:\FOPCS

08/02/2019  04:26 PM    <DIR>          .
08/02/2019  04:26 PM    <DIR>          ..
08/02/2019  04:25 PM               412 FirstProgram.cs
08/02/2019  04:26 PM             3,584 FirstProgram.exe
               2 File(s)          3,996 bytes
               2 Dir(s)  701,679,099,904 bytes free

d:\FOPCS>
```

https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/compiler-options/command-line-building-with-csc-exe

Running

- The compilation process will produce an executable file – FirstProgram.exe
- We can call this executable file from the command prompt to run our program
- We should see the text that we put in our program printed on the screen.



Syntax Errors

- Syntax errors happens when the compiler cannot correctly understand our program e.g. we mistype something
- Let's try to put in some typo error.

```
static void Main(string[] args)
```

```
static void main(string[] args)
```

Developer Command Prompt for VS 2017

```
d:\FOPCS>csc FirstProgram.cs
Microsoft (R) Visual C# Compiler version 2.8.3.62923 (7aafab56)
Copyright (C) Microsoft Corporation. All rights reserved.

error CS5001: Program does not contain a static 'Main' method suitable for an entry point

d:\FOPCS>_
```

Syntax Error

- If we replace `WriteLine` to `WriteWine`

Developer Command Prompt for VS 2017

```
d:\FOPCS>csc FirstProgram.cs
Microsoft (R) Visual C# Compiler version 2.8.3.62923 (7aafab56)
Copyright (C) Microsoft Corporation. All rights reserved.

FirstProgram.cs(14,21): error CS0117: 'Console' does not contain a definition for 'WriteWine'
FirstProgram.cs(15,21): error CS0117: 'Console' does not contain a definition for 'WriteWine'

d:\FOPCS>_
```

- The compiler will try to infer the location of the syntax error
  - We use this information to find the error and fix it
  - Sometimes the location inferred is not accurate – we need to analyse our own source code to find the root problem

Common Beginner's Syntax Errors

- Using a wrong case e.g. Main become main
- Forget to terminate a statement with semi-colon
- Typo on class names or method names
- Mismatch curly braces, brackets or quotes
- Use the short name of a class without importing the namespace with "using" directive

Summary

- We have covered C# programming language and how to create a simple C# program
- Syntax in programming language have to be followed very strictly
  - Otherwise you will get syntax errors and your program won't work

Reference

- C# Language Reference
  - https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/
- C# Programming Guide
  - https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/
- .NET API Browser
  - https://docs.microsoft.com/en-gb/dotnet/api/?view=netframework-4.7.1
  - Reference to find out the methods that can be used in .NET class libraries